Probing the Performance of the Edinburgh Bike Sharing System Using SSTL

Probing the Performance of the Edinburgh Bike Sharing System using SSTL

Justin Noah Kreikemeyer
Jane Hillston
justin.kreikemeyer@uni-rostock.de
jane.hillston@ed.ac.uk
University of Edinburgh, School of Informatics
Edinburgh, United Kingdom

Adelinde Uhrmacher
adelinde.uhrmacher@uni-rostock.de
University of Rostock, Faculty of Informatics and Electrical Engineering
Rostock, Germany

ABSTRACT

Bike sharing systems are a popular form of sustainable and affordable transport that has been introduced to cities around the world in recent years. Nevertheless, designing these systems to meet the requirements of the operators and also satisfy the demand of the users, is a complex problem. In this paper we focus on the recently introduced bike sharing system in the city of Edinburgh and use data analytics combined with formal modelling approaches to investigate the current behaviour and possible future behaviour of the system. Specifically we use a spatio-temporal logic, SSTL (the signal spatio-temporal logic), to formally characterise properties of the captured system, and through this identify potential problems as user demand grows. In order to investigate these problems further we use the CARMA modelling language and tool suite to construct a stochastic model of the system to investigate possible future scenarios, including decentralised redistribution. This model is parameterised and validated using data from the operational system.

CCS CONCEPTS

• **Computing methodologies** → **Modeling and simulation**; *Agent / discrete models*; • **Theory of computation** → **Modal and temporal logics**; • **Applied computing** → Transportation.

KEYWORDS

SSTL; model checking; bike-sharing; latent demand; decentralised redistribution; discrete simulation; agent-based simulation

ACM Reference Format:
Justin Noah Kreikemeyer, Jane Hillston, and Adelinde Uhrmacher. 2020. Probing the Performance of the Edinburgh Bike Sharing System using SSTL. In *Proceedings of the SIGSIM Principles of Advanced Discrete Simulation (SIGSIM-PADS '20), June 15–17, 2020, Miami, FL, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3384441.3395990

SIGSIM-PADS '20, June 15–17, 2020, Miami, FL, USA

ACM ISBN 978-1-4503-7592-4/20/06…$15.00
https://doi.org/10.1145/3384441.3395990

1 INTRODUCTION

In the last decade there has been a significant increase in the number of bike-sharing systems (BSS) deployed around the world [11]. These systems are conceptually very simple: a number of bike stations are distributed over a geographical region or city. Each station has the capacity to store bicycles and some form of authorised release system which allows users to take a bike, make a journey and return the bike to another station. In most cases users pay for the service, but a variety of different funding models are deployed, ranging from a one-off membership fee, to a per minute charge per journey. From the operator's point of view, in addition to the initial investment in the bikes and station infrastructure, there are also operational costs associated with bike redistribution. This is because in most cities the BSS does not achieve a natural equilibrium, leading to imbalance in the system. For example, users are much more reluctant to make journeys uphill than down. If left unaddressed this imbalance can lead to user dissatisfaction as they are unable to make a planned journey, either because a bike is not available to start their journey or because a parking slot is not available in the destination station to allow them to complete their journey. Bike redistribution is expensive, and also decreases the sustainability of the BSS which is often offered as a "green" alternative form of transport.

Several research problems associated with BSS have been explored in the literature, including policy design [21, 32], intelligent bike redistribution [8, 16, 26, 36], inventory level optimization [9, 30, 34], and user journey planning [14, 38]. In this paper we focus on the redistribution problem and investigate a decentralised approach to bike redistribution based on user incentives. The strategy we use is similar to the ones found in [1, 33] as we use the same parameters, alternate station search radius and user cooperation, to abstract from a specific implementation. We evaluate this strategy on a significantly smaller system where stochastic effects have a high influence. Our main contribution is to provide a novel perspective by using spatio-temporal logical expressions to evaluate the performance. Together with visualisation on a geographical map, this allows for identification of low-performing stations within the system at first sight.

Our approach is a combination of data analytics, spatio-temporal model checking and stochastic modelling. Using data from the recently installed Edinburgh BSS we use spatio-temporal model checking to interrogate the data and investigate problems of user dissatisfaction arising from thwarted journeys. Desirable properties of the system, such as *"there will always be a bike and slot available*

at the station within a specified time interval" or undesirable ones such as *"the station will always eventually be full (empty) over the course of a day"*, can be expressed in the logic SSTL (the signal spatio-temporal logic) and checked automatically against logging data. This analysis can reveal when there is potential dissatisfaction amongst users because it will not be possible to start or complete desired journeys. In order to investigate this further we construct a stochastic spatio-temporal model of the system. Unlike the data, the model allows us to make modifications to the system and test the impact on the operation of the BSS as well as levels of user satisfaction/dissatisfaction. We use the CARMA (Collective Adaptive Resource-sharing Markovian Agents) modelling language and associated tools to construct our model. In this model stations, bikes and users are modelled as interacting agents. Using trajectories generated from the model now, rather than the operational system, we use the same SSTL formulas to interrogate the behaviour of the modified BSS and draw conclusions about which measures are most likely to be beneficial to both users and operators.

2 BACKGROUND

In this section we introduce some background information about the Edinburgh BSS, and fundamental concepts and notation for the logic SSTL and the modelling language CARMA, which we deploy in our investigation of the system.

2.1 Edinburgh Cycle Hire

The Edinburgh Cycle Hire (ECH) system was introduced into the city of Edinburgh in September 2018 with about 15 stations. Since then it has experienced significant growth to just under 90 stations in December 2019. During that time, the operator periodically added stations in and around the city and subsequently removed some based on the new usage statistics. This results in a high variance in station number and positions, even within the same month, and makes a study over multiple months very difficult. We chose one month, August 2019, since the festival period in Edinburgh during this month puts an interesting load on the scheme. The population of the city usually doubles as people visit the city for a multitude of events, imposing major challenges for transport providers.

Our analysis is based on two historic datasets with data collected from 1st to 31st August 2019. The first ("Trips Data") is accessible on the ECH website[1] [10]. It consists of one entry for each trip showing start time and location, end time and location, and duration. Only trips with a duration greater than one minute are included. This set does not include the capacities of the stations, which is crucial information as it limits the performance. Thus, an additional dataset ("Availability Data") was requested from the operator of ECH. This contains the number of available bikes and free slots for each station measured every minute. It is important to note that the Trips Data does not contain the redistributions done by the operator, while the Availability Data does contain them implicitly as changes of the number of available bikes/free slots.

[1] As of January 2020, the last day is missing for each month. For this analysis, the operator provided the complete sets separately.

2.2 Signal Spatio-Temporal Logic (SSTL)

The spatio-temporal logic, SSTL, was introduced in [27] and interested readers can find more detail in that reference. Here we aim to present the main features of the logic and an intuitive understanding of how to read and understand formulas in the logic. SSTL is a spatial extension of Signal Temporal Logic (STL) [25], a temporal logic suitable for describing properties of real-valued signals. The signals are defined by inequalities over attribute values associated with locations. Here we assume that the signals are based on the number of bikes or slots available in a BSS. Thus the underlying model is a spatial population model which is comprised of a number of locations, organised as a finite weighted undirected graph $G = (L, E, w)$, where L is a set of nodes, E a set of connections or edges and $w : E \to \mathbb{R}_{\geq 0}$ is the *cost* or weight associated with an edge. In our case the bike stations are the locations, and the weights are the distances between them. Each location is assumed to have a finite set of possible states recording the number of bikes/slots available, and the state of the system at any given time is the vector of population values, one for each location. Such a spatial model with values associated with each location can be constructed directly from data or from a spatio-temporal model constructed in a formalism such as CARMA. When we consider the evolution of the model over time, we can construct a *spatio-temporal trajectory*, σ, recording the state of the system at each time step.

SSTL allows us to express properties of such a spatial population model in a formal way. Its syntax is given by:

$$\varphi ::= \mu \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \, \mathcal{U}^{[t_1,t_2]} \, \varphi_2 \mid \; \diamondsuit_{[w_1,w_2]} \, \varphi \mid \varphi_1 \mathcal{S}_{[w_1,w_2]} \varphi_2$$

The SSTL *atomic proposition* (or basic property) μ is of the form $\mu \equiv (f \geq 0)$, $f : \mathbb{R}^n \to \mathbb{R}$, an inequality on expressions over population counts, given in the spatio-temporal trajectory. *Negation* $\neg$ and *disjunction* $\vee$ are the standard boolean operators and $\varphi_1 \, \mathcal{U}^{[t_1,t_2]} \varphi_2$ is the *bounded until* operator. This temporal operator is used to verify that the property φ_2 will be satisfied at some time instant in the interval $[t_1, t_2]$ and that at all preceding time instants φ_1 holds. SSTL introduces two spatial operators: the *bounded somewhere* operator $\diamondsuit_{[w_1,w_2]}$ and the *bounded surround* operator $\mathcal{S}_{[w_1,w_2]}$, with w_1, w_2 real values, $w_1 \leq w_2$. The *bounded somewhere* operator requires that the property φ holds in a location reachable from the current one with a cost w, $w \in [w_1, w_2]$. The operator *bounded surround* describes the property of being surrounded by a φ_2-region, while being in a φ_1-region: the formula $\varphi_1 \mathcal{S}_{[w_1,w_2]} \varphi_2$ is true in a location l, if l belongs to a set of locations A where φ_1 holds, such that its external boundary $B^+(A)$ contains only locations satisfying φ_2. The external boundary of a subset of locations A is defined as $B^+(A) := \{l \in L \mid l \notin A \ \wedge \ \exists l' \in A \text{ s.t. } (l, l') \in E\}$. Moreover, the locations in the $B^+(A)$ have to be reached from location l with a cost w, $w \in [w_1, w_2]$. We will also find it convenient to use the following derived syntax:

$$\mathcal{F}^{[t_1,t_2]}\psi := T \, \mathcal{U}^{[t_1,t_2]}\psi \qquad\qquad \mathcal{G}^{[t_1,t_2]}\psi := \neg\mathcal{F}^{[t_1,t_2]}\neg\psi$$

where $\mathcal{F}$ denotes the *eventually* operator and $\mathcal{G}$ denotes the *globally* operator.

Examples of SSTL formulas are provided in Section 3. SSTL is equipped with both boolean and quantitative semantics; the former returns the value true/false ($\mathbb{B} = \{T, F\}$) depending on whether

the observed trajectory satisfies the defined SSTL formula or not. The latter semantics return a measure of the robustness of the satisfaction or dissatisfaction. The formal definition of the boolean semantics of a SSTL formula φ is rather technical and we refer the interested reader to [27] for details. In our current work we use only the boolean semantics.

Monitoring algorithms have been defined to evaluate the validity of SSTL properties, given a spatio-temporal trajectory, working inductively bottom-up on the parse tree of the formula. To make the verification procedure tractably computable, the time-domain has to be discretised, giving as output a piece-wise constant approximation of the result.

In the study of stochastic systems we are generally interested in evaluating the probability that given properties are satisfied; a commonly used approach consists of estimating these values using statistical methods on a set of trajectories [13]. Therefore, given a SSTL property φ, we shift the analysis from a single trajectory σ to a set of trajectories Σ, assigning to each trajectory a truth value, according to the boolean semantics. Let $\beta(\sigma, l, t, \varphi)$ denote the truth value of φ with respect to the trajectory σ at location l and time t. We can estimate the satisfaction probability p^* of the formula φ. We define $\mathcal{P}_\beta$ over the set of trajectories Σ, in terms of β:

$$p^* = \mathcal{P}_\beta(\Sigma, l, t, \varphi) = \frac{|\Sigma_\top|}{|\Sigma|}$$

where $|\Sigma_T| = \{\sigma \in \Sigma \mid \beta(\sigma, l, t, \varphi) = T\}$.

2.3 CARMA

CARMA is a novel modelling agent-based language, designed to study the collective behaviour of multi-agent systems acting and interacting in an environment [24]. It is a stochastic process algebra, meaning that the primitives of the language are agents (or "processes") and actions, and actions are assumed to have an associated duration which is governed by an exponential distribution. In CARMA the model is formed as a composition of components (agents) interacting in an environment, where each component consists of one or more process and a store of attribute values. Each process can undertake a set of actions either independently or as communication with other processes in other components.

The language offers a rich set of communication primitives, and exploits attributes to enable attribute-based communication [3]. For example, for components that have a location attribute, attribute-based communication allows communication to be limited to components that are co-located [17], e.g. in our scenario a user can only take a bike from a station if they are in the same location.

Specifically, CARMA supports both unicast and broadcast communication, and permits locally synchronous, but globally asynchronous communication. Distinct predicates (boolean expressions over attributes) associated with senders and potential receivers are used to filter possible interactions. Thus, a component can receive a message only when its store satisfies the target predicate. Similarly, a receiver also uses a predicate to identify accepted sources. Thus a user can only return a bike to a station whose attribute indicates that there is a free slot available. Predicts also act as guards on actions. For example, in our model this can be used to allow dissatisfied users to leave the system rather than wait indefinitely.

In CARMA, the environment captures the physical environment and mediates agent interactions. Specifically the environment is responsible for setting the rates at which actions are performed, and probabilities of receiving a given message. So in our model, the environment "knows" the distance between stations and thus can assign a likely journey time for any particular trip. The environment is also responsible for generating components and setting initial attribute values.

A CARMA system has a semantics that gives rise to an underlying Continuous Time Markov Chain (CTMC). This is not generated explicitly but the semantics provides the basis for Monte Carlo simulation of the CTMC, based on the Gillespie Algorithm [15]. There is a Java implementation which carries this out automatically [22]. For our model, this generates to a spatio-temporal trajectory. This trajectory records for each time step, for each location, the state of the system in terms of the number of components of each type at that location. Just as with the logging data from the BSS system, these trajectories can be queried using SSTL formulas.

3 USING SSTL FOR DATA ANALYSIS

In this section we explain how we use SSTL to investigate the behaviour of the system using the logging data available for the ECH system. This data records the number of bikes/slots available at each station at each time slot, where a slot consists of a minute. The performance of the system is evaluated for an average weekday of $T = 1440$ minutes (24 hours). The SSTL formulas that we used are closely related to those presented in [28]. Furthermore the same spatio-temporal variables B (number of available bikes) and S (number of free slots) are used.

First indicators of performance issues are stations without available bikes (empty stations) and stations with no free slots (full stations). These characteristics are captured by the formulas φ_{empty} and φ_{full}. In simple terms these state: *"The station will eventually be empty/full during the day."* The parameter T represents the final time.

$$\varphi_{empty} = \mathcal{F}^{[0,T]}(B = 0) \qquad \varphi_{full} = \mathcal{F}^{[0,T]}(S = 0)$$

Another way to measure the performance of a BSS is to combine φ_{empty} and φ_{full} and look at so-called problematic stations. These are stations that have neither a bike nor a slot available [12]. φ_{prob} captures unproblematic stations: *"Within the specified interval $[t_s, t_e]$, the station always has a bike and a free slot available"*. Since the observed behaviour of the system varies according to the time of day, t_s and t_e are varied to represent four six-hour periods.

$$\varphi_{prob} = \mathcal{G}^{[t_s,t_e]}\{(B > 0) \wedge (S > 0)\}$$

The preceeding measures are primarily of interest to the operator of the system. From a user's perspective, it is important to know whether one can always find a bike and a free slot within certain limits. The formula below states: *"There is always a bike and a slot available at distance d"*:

$$\varphi_{dist} = \mathcal{G}^{[0,T]}\{\diamondsuit_{[0,d]}(B > 0)\ \diamondsuit_{[0,d]}\ (S > 0)\}$$

At $d = 0$ this formula describes an unproblematic station, that always has a bike and a free slot available. In our analysis the parameter d is varied in 50m steps up to 500m reflecting different

assumptions about users' willingness to walk or ride to another station in order to achieve their objective.

A user may also be interested to know whether waiting at a specific station will help them retrieve or return a bike. φ_{time} describes that *"after t minutes of waiting there is always a bike and a slot available"*:

$$\varphi_{time} = \mathcal{G}^{[0,T]}\{\mathcal{F}^{[0,t]}(B > 0) \wedge (S > 0)\}$$

As an initial investigation and in order to establish a baseline, we first evaluate the performance of the system with respect to these formulas as it is captured in the data, including redistributions. An overview of the process is shown as "Path 1" in Figure 1. For this, the ECH availability data is used. The dataset is split into individual (week-)days[2], resulting in 21 trajectories, each describing the availability of bikes and slots for all stations at each minute of the day. Those can almost directly be used as spatio-temporal signals by using the values of the availability fields for the variables B and S. For the spatial graph G, the stations form the vertices and the edges connect all pairs of distinct stations. The edge weight is set to be the straight line distance between stations and calculated using the Haversine Formula [6]. Since the data has a granularity of one minute and starts at 0, the final time T is set to 1439 minutes. The details of the data-conversion and the code used for evaluation can be found on GitHub [18].

Loading the graph and evaluating the formulas on the trajectories is achieved using the jSSTL java library [23, 29]. This Java library implements classes and algorithms to allow for the specification and evaluation of SSTL formulas. In our case, we use it to implement the previously described formulas and calculate the satisfaction probability p^* over all 21 trajectories (see the end of Section 2.2). The results can be seen in Figure 2, 3 and 4. Please note, that the operation of the system changed since our evaluation. The following key observations can be derived for August 2019:

- Stations in the south of the city show a greater tendency to be empty, some with a probability higher than 50% and many with a probability close to 50% (Figure 2 a).
- From the data there does not appear to be any significant problem with full stations (lack of slots, Figure 2 b).
- Even when the search radius is expanded to 300 meters, it is not guaranteed to find a bike at any time (Figure 3 a-c).
- Waiting at a station has almost no effect on availability (Figure 3 d-f).
- Over time most stations become more problematic. Some regain bikes at the end of the day. This can be explained by commuting customers and redistributions by the operator (Figure 4).

To conclude, the captured performance of the system leaves room for improvement. Moreover, non-zero probabilities for φ_{empty} and φ_{full} mean that the corresponding station may be empty/full at some point. During that time, no new journeys or returns would be possible. Thus, some attempted user interactions are missing in the provided data, masking the actual demand. This is a common effect when predicting demand (see also [2, 19, 30]), and we will show how such latent demand can be incorporated into analysis in Section 5.

[2]Data corresponding to Saturdays and Sundays is discarded, as this is atypical.

4 BUILDING A BIKE SHARING MODEL IN CARMA

In Section 3, the performance of the observed system was evaluated based on the Availability Data. To introduce additional performance measures and carry out experiments to explore future scenarios, a stochastic model was developed using the CARMA language. The complete code with usage instructions is available on GitHub [18].

We base our model on the model introduced in [24] and elaborate on it. The lower section of Figure 5 shows a diagram of the components and their interactions. The model is composed of three types of components: *User*, *Station*, and *Spawner*. The *Spawner* generates users at a specific *Station* with the *spawn*-action based on a time-inhomogeneous arrival process. *User*s first choose a destination based on the current time and origin. They then attempt to retrieve a bike from their origin *Station* via the *get_bike*-action, ride it for some time (*move*) and return it (*return_bike*) at their destination. The *Station* synchronises with *User*s on the actions *get_bike* and *return_bike* and updates the number of available bikes in its local store accordingly, but only if there are enough bikes/slots available. As initial configuration, one representative *Station*-component for each station in the real system and one *Spawner*-component for each station is placed in the environment.

For parameterisation we make use of the Trips and Availability Data described in Section 2.1 to represent the system during an average weekday. A summary of this phase is depicted in the upper part of Figure 5. To clean the data, first all weekends are removed, as those have a different usage pattern. In fact, due to the morning and evening rush hour, when looking at the number of trips per hour of day, weekdays exhibit an "M"-shaped curve and weekends exhibit a bell-shaped curve (see Figure 6), as already discovered in [4, 20, 31]. Subsequently, all stations with less than 15 departures or arrivals in the whole month are excluded. This captures those stations that were removed at the beginning or only introduced at the end of August. Also all trips with a duration greater than the 99-th quantile are removed to eliminate outliers. The cleaned Trips Data is split into two different sets[3]: one used for parameterisation ("Training Data") and one for validation ("Validation Data") by randomly picking 5 days, so that one of each of the five weekdays is included in the Validation Data to ensure unbiased validation. This way, the original Trips Dataset containing 17478 journeys is reduced to 11211 journeys, of which 2264 are used for validation and 8947 are used for training.

The arrival process for a single station is then generated as a piecewise constant function from the Training Data by counting the number of departures during each hour of the day and taking the average over all days. This value is then converted to a rate and used for the *spawn*-action. To determine the rate for the *move*-action, the average duration for a journey between each pair of stations is used. Destinations are determined based on the distribution of the destinations for each station at every hour of the day. The stations are derived from the Availability Data, with their initial number of available bikes calculated as the median number of available

[3]To validate the model, independent data (e.g. data from August 2018) would be the best choice. However the ECH system was not yet introduced during that period. We thus decided to use an approach commonly used in machine learning and split the data to test the predictive capabilities of our model.

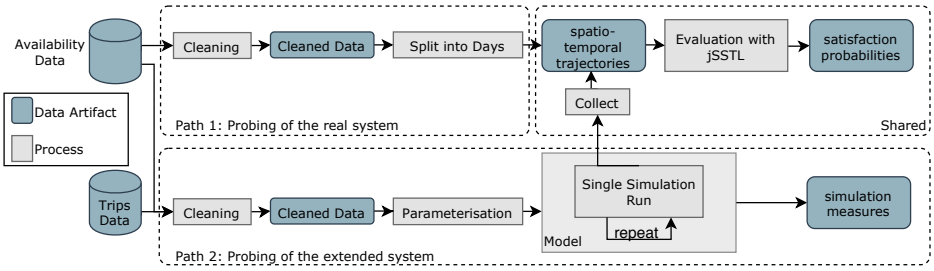


Figure 1: Diagram of our approach. Path 1 is used in Section 3 and Path 2 in Section 5.

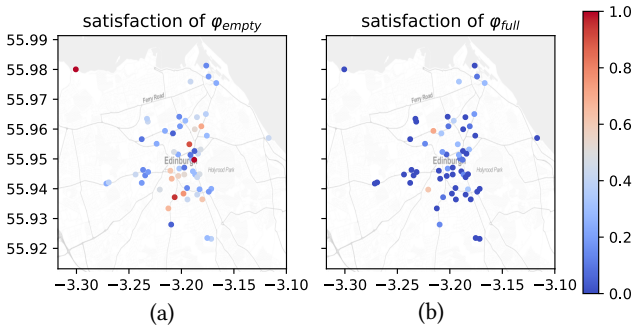


Figure 2: Satisfaction of φ_{empty} (a) and φ_{full} (b) for the different stations, which are shown as coloured dots on a map of Edinburgh. For this figure and all following similar figures, the values for the x- and y-axis are given in degrees longitude/latitude and the colour represents the satisfaction probability of the given formula in percent from 0% (blue) to 100% (red). A lower value at a location means a better performance. For instance, the blue dot in the top left of (b) means, that φ_{full} has a near 0% chance of being true for this location, which means that the corresponding station is unlikely to be full at some time at a given day. Map tiles by Stamen Design (http://stamen.com/), under CC BY 3.0 (http://creativecommons.org/licenses/by/3.0). Map data by OpenStreetMap (http://openstreetmap.org/), under ODbL (http://www.openstreetmap.org/copyright).

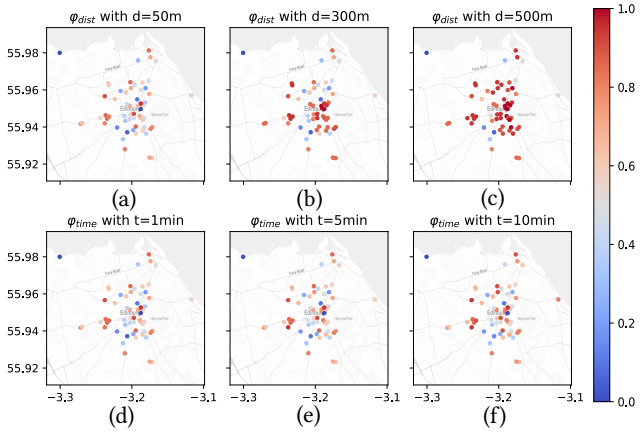


Figure 3: Satisfaction of φ_{dist} for $d \in \{50, 300, 500\}$ (a-c) and φ_{time} for $t \in \{1, 5, 10\}$ (d-f). A higher value indicates a better performance.

bikes in the inclusive interval between 12 and 2 am. The rates for the *get_bike* and *return_bike* actions are set to a very high value, meaning they happen almost instantly, as the time it takes to check out a bike is already included in the trips duration. Both of them are multiplied by the number of users that currently want to retrieve (or return) a bike, simulating parallel retrievals and returns. In the model, one time unit represents one minute of physical time, e.g. a rate of 1 means, that the corresponding action happens on average once every minute.

Several assumptions are made to keep the model simple. First, it is assumed that each weekday experiences roughly the same load. This is only true to a certain extent as [4, 20] show, but it is reasonable to assume when focusing on average performance. Factors such as weather and season also play a large role, but are not taken into account due to a lack of data. Furthermore, the arrival rate is assumed to be hourly piecewise constant, as is the distribution of destinations. Testing with larger constant intervals for the destination distribution revealed that the preferred destinations differ significantly for each hour, making larger intervals inaccurate.

Several values in the simulation are measured at each sampling point (each minute). This includes the cumulative number of overall bike retrievals and the number of available bikes at each station. To produce a spatio-temporal signal from the model outputs, the values of the two spatio-temporal variables B (number of available bikes) and S (number of free slots), introduced in Section 3, are stored for each station at each time, creating one trajectory per replication. These can then be used for evaluation through statistical model checking as described in Section 3.

4.1 Validating the Model

Validation of the model is done by comparing the measures (see Figure 5) of the simulation, parameterised as in Section 4, with the validation data and the training data. Since some of the stations changed capacity during August 2019, it is necessary to disable the capacity- and availability-limit of stations for validation, because the real data might include trips that exceed those limits. 10000 replications are performed to achieve an average 99% confidence interval of ±0.068. First the mean over the time series for each replication is calculated. Based on these means the confidence interval is calculated.

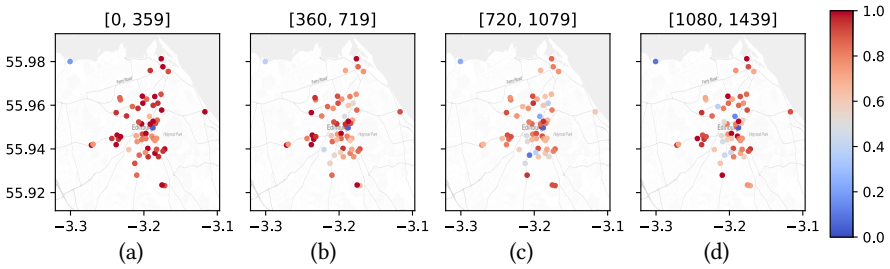


Figure 4: Satisfaction of φ_{prob} for different intervals $[t_s, t_e]$. A higher value indicates a better performance.

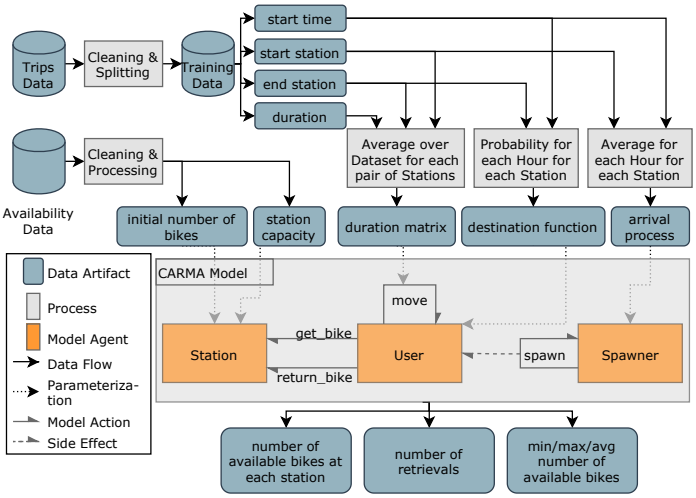


Figure 5: Diagram of the parameterisation process and CARMA model, giving a detailed view of "Path 2" in Figure 1.

At a high level, summary statistics such as the number of bike retrievals during each hour can be considered, as seen in Figure 6.

At a more detailed level, the time series of the number of available bikes resulting from the simulation can be compared to the time series generated from the training/validation data. To achieve this, we calculate the mean squared error (MSE) at each hour. The maximum of the resulting series is taken and used as an error value for each station. Table 1 shows the minimum, maximum, mean and median of those values over all stations. As might be expected, the error of the model output relative to the Training Data is very small. The error in relation to the Validation Data is also reasonable, with the median of the described error measure being approximately one bike. In spite of this, one can observe a significantly higher value for the maximum. Keep in mind that, as a result of squaring, this does not reflect the absolute difference in number of bikes.

As mentioned earlier, the BSS in Edinburgh is still under development and is relatively small. The small size means that external influences, like the weather conditions on a particular day, can have a large impact on the system, thus resulting in a high variance for the number of available bikes and also number of trips on any given day. Additionally, the varying redistributions of the operator introduce additional variance. In this study we focused on an average weekday, to establish the methodology and demonstrate its potential for more in depth studies. With this in mind, it can be concluded, that the predictive capabilities of the model are also sufficient for its purpose.

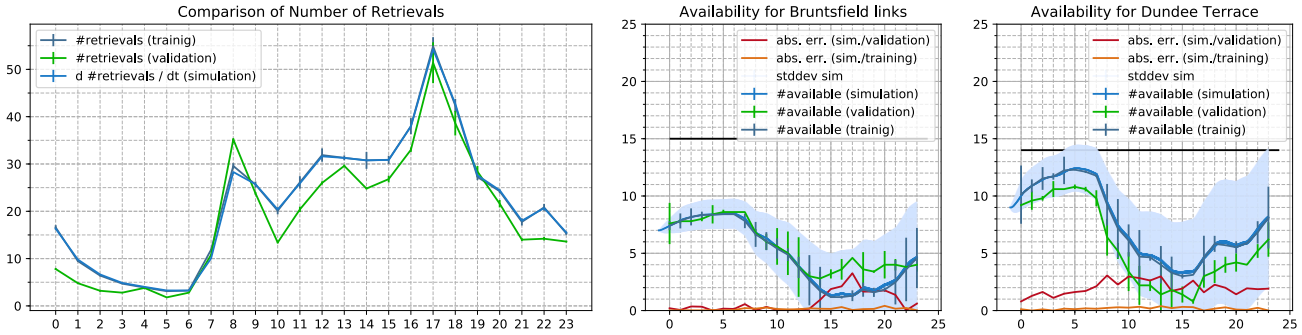
4.2 Extensions to the Model

In Section 3 we saw that the performance of the system is acceptable, but not optimal, even with the current redistributions. In order to get a better understanding of how the users perceive the system, a user policy, that dictates their behaviour in case they don't find a bike or slot, is introduced and their dissatisfaction is measured. With this the dissatisfaction levels resulting from the currently operating system are established. We then modify the model to incorporate an alternative to static repositioning — an incentive-based, decentralised redistribution strategy. This is also evaluated with respect to user dissatisfaction and the measures identified previously. Finally, a measure for the redistribution effort is presented to quantify the effect of the decentralised redistribution strategy from the operator's perspective compared with static redistributions.

4.2.1 User Policy. In the initial model, users wait until a bike (or slot) is available. Contrastingly, in a real system users would wait only for a certain amount of time or try to find a bike/slot at a nearby station. As seen in Section 3, waiting (for a reasonable time) has almost no effect on availability. Thus we now assume that users

Table 1: Error measures for the model. The maximum MSE of the hourly fill level in the simulation in relation to training and validation data is calculated for each station. Then, mean, median, max, and min are applied across all stations.

Dataset	min. MSE	mean MSE	median MSE	max. MSE
Training	0.001	0.015	0.008	0.123
Validation	0.042	1.686	0.960	24.944



(a) **(b)** **(c)**

Figure 6: (a) Shows the number of retrievals in the simulation (light blue) compared to training (dark blue) and validation data (green). Notice the "M"-shaped form of the curve. (b) and (c) show the number of available bikes throughout the day for the stations "Bruntsfield Links" and "Dundee Terrace", the horizontal black bar shows their capacity. The colours are the same as before, with the thickness of the light blue line showing a 99% confidence interval and the light blue shadow showing the standard deviation of the runs. The error bars show the variance. The error for validation/training is shown in red/orange.

will seek a bike in nearby stations until they give up and use another means of transport.

To reflect this in the model users that fail to retrieve a bike will walk for an exponentially distributed time with a mean of 5 minutes to a randomly selected nearby station, defined as any station within 350 metres of the original station. Users will never visit a station twice, and users leave the system after three unsuccessful attempts. At the start of a journey, this means the user will just use another means of transport. However when unable to make a return, a user will leave the system with the bike because there is no place to put it. In the model each unsuccessful attempt to return/retrieve a bike results in the internal counter of the respective number of failures being increased in the User component. For example, a User with dissatisfaction level 0 for retrieving bikes, experienced no difficulties getting a bike, while a user with dissatisfaction level 2 for returning had to visit two other stations before being able to return the bike.

For the measure to be easily comprehensible, the expected value for the number of such "hops" a user has to take is calculated as follows:

$$D(t) := \frac{\sum_{l=0}^{3} d_{get}(l,t) * l}{\sum_{l=0}^{3} d_{get}(l,t)} + \frac{\sum_{l=0}^{3} d_{ret}(l,t) * l}{\sum_{l=0}^{3} d_{ret}(l,t)}$$

with $d_{get}(l,t)$ and $d_{ret}(l,t)$ describing the number of users with dissatisfaction level l at time t for retrieving and returning bikes respectively. Note that the maximum value for D is 6 rather than 3 (the maximum value for d).

4.2.2 Incentive-Based Redistribution. In order to ensure user satisfaction, the system needs to be as balanced as possible. One way to achieve this, is by using a fleet of trucks to redistribute the bikes during the night when the usage is low, commonly referred to as static or centralised repositioning. Another approach is to use a dynamic decentralised strategy and divert the users to slightly different stations when it would improve the balance in the system, by providing an incentive to them. In this way, bikes can be returned to/retrieved from the stations that need them the most while also saving the fuel costs and emissions that result from using trucks for centralised redistribution. Additionally, the probability that customers will find a bike/slot can be increased. Such strategies have already been evaluated in the context of other systems, for example in London [1] or New York [7, 31]. However, the system in Edinburgh differs in that it is magnitudes smaller. It is thus interesting to see whether the same positive effects can be observed. Similar to [1], we do not specify the kind of incentive used, but introduce a user cooperation factor, that simulates the degree of takeup. A detailed survey can be found in [37].

In every case, such a strategy depends on a real time heuristic for the future number of available bikes at a specific station. To determine an estimate of the future number of bikes available t hours ahead of the current time, the following formula is used:

$$a_{fut}(t_0, s) = a(s) - d(t_0, s) + r(t_0, s)$$
$$a_{fut}(t_0 + t, s) \approx a_{fut}(t_0 + t - 1, s) - d(t_0 + t, s) * 0.75^t + r(t_0 + t, s)$$

where $a(s)$ is the current number of available bikes at station s, t_0 is the current hour, and $d(t,s)$ and $r(t,s)$ are the expected number of checkouts (demand) and returns at hour t at station s. The demand is determined from the arrival process and the number of returns is updated every time a user decides on a destination. To account for the limited accuracy with increasing time, the demand is discounted. Because the horizon of the returns is naturally limited, a discount is not necessary. This is a simplification. In the real world the value will be the sum of an estimate based on the historic (and probably also weather) data and the current known trips from the incentive strategy.

Exploring the data for Edinburgh revealed that achieving a certain optimal fill level for each station almost balances the system for the whole day. Accordingly the final heuristic p will be the estimated difference from that fill level at the end of the current day:

$$p = a_{fut}(\lceil t_0/24 \rceil * 24, s) \quad - \quad \text{desired fill level of station } s$$

For stations that need bikes, $p < 0$ and for stations that need slots $p > 0$. Other systems may use the same strategy by adjusting the time window, desired fill levels and discount factor.

By taking the minimum/maximum over neighbouring stations, this prediction can now be used to determine which station will need a return/retrieval the most. It is also important to consider current constraints of the “losing” station to sustain a decent performance. This means that stations should not agree to reroute a trip when their fill level is below or above a certain threshold.

To realize such a strategy in the real world, a user may use an app that will tell her in advance, based on the planned trip, which station for return and retrieval will be optimal for the system and what the incentive will be. Obviously, not all users will cooperate. In the model, this fact is represented by a cooperation factor that determines the percentage of cooperative customers. Another important factor is the radius for the neighbourhood in which alternate stations are considered. This can have a major influence on the performance as shown in [1]. In Section 5 we determine the influence of cooperation and neighbourhood radius for the ECH system.

4.2.3 Redistribution Effort. To quantify the number of redistributions that have to be done, a measure for the displacement of the fill-levels is introduced. This “Redistribution Effort” is calculated by summing up the absolute difference between the actual and optimal number of available bikes for each station after each day (or 1440 time units). The resulting value indicates how many bikes need to be moved to rebalance the system. Additional factors, such as travelling distance for a truck, are not taken into account.

5 PROBING THE PERFORMANCE OF THE EXTENDED SYSTEM

In the previous Sections 3 and 4 the performance of the system as-is was evaluated using SSTL and a model was built to make extensions possible. In this section, the system is evaluated again using the model, exploring the possible future performance in different scenarios, as displayed in Table 2. Throughout this section, the same abbreviations as in the table are used within the brackets to refer to the corresponding row. These are: LD for "latent demand", Opt for "optimal initial fill levels" and Inc d c for the "incentive strategy with maximum distance d and user cooperation factor c".

The experiments were carried out on a Intel Xeon E5-2690 processor, clocked at 3.0 GHz and the replications were distributed across 8 threads. One might expect some experiments to take longer than others as the incentive strategy requires additional computations, but these impacts where found to be minor. On average, the experiments took 15 minutes to complete all replications. The main factors, that impact the scaling of this approach, are the end time of the simulation, the number of simultaneous users and the number of stations. At the moment, the number of stations and their parameters that can be simulated using CARMA is limited by the maximum method size, which is imposed by Java. The evaluation of the SSTL formulas took another 15 minutes for 1000 traces. The time it takes to evaluate a given formula increases with the number of used operators. For instance, the formula φ_{prob} takes about 0.11 seconds while φ_{empty} takes around 0.006 seconds to evaluate on a single trace. It should also be noted that the traces require a significant amount of disk space (4.8 GB for 1000 traces), as one trace is generated per replication.

First, a baseline for the performance without redistributions is established. This baseline is then corrected for latent demand. Eventually, the impact of the incentive strategy proposed in Section 4.2.2 is measured for different parameters and compared to an optimal static redistribution strategy. The simulation is run for 3 days ($T = 4319$) to also catch longer term effects and 1000 replications are performed for each experiment [18].

In order to get a view without redistributions on the system, the extended model is parametrised as described in Section 4 (**Baseline**). One can see, that this baseline is similar to the performance measured in Section 3. An example of this is shown in Figure 7. To estimate the latent demand, we use a strategy similar to [2]. Each period where a station was empty for a full hour is considered. Next, we calculate the average number of trips for this period, only including those where the station was not empty. Finally we randomly generate some trips that correspond to the average behaviour in that period for that station. This way, we get an idea of the thwarted journeys the Trips Data is missing and the total number of trips is increased by 3794, which is a 22% increase to the 17478 journeys mentioned in Section 4 (**Baseline LD**).

It was discovered, that the initial distribution of bikes gained from the Availability Data is far from optimal and filling the stations with the correct number of bikes can almost balance the system for a full day (**Baseline LD, Opt**). To calculate the optimal number of available bikes, a simple hill climbing algorithm is used on the Trips Data with the goal to keep the fill level throughout a day three bikes above zero and three bikes below capacity.

The three previous experiments did not include any redistribution. Next, the incentive strategy is evaluated with different parameters for user cooperation and station radius (**Inc Radius CooperationFactor**). The fill levels start at their optimal value to simulate a former repositioning. The results indicate that a purely incentive based redistribution is not feasible in the long run, even for a large distance and high cooperation factor. However, there is a positive impact in the number of redistributions that have to be made, with most of the configurations cutting the redistributions

in half and some making a redistribution every two days (on average) feasible. As expected, it also increases the number of trips made. Considering that, due to high fuel costs, static repositioning is likely to be more expensive than the incentive-strategy, this strategy could be used as a cost saving measure. Table 2 quantifies the effects described. Those numbers can also be used to calculate costs based on how the incentive is realised. Additionally, the impact of the strategy scales very well with user cooperation, keeping most of its positive impacts even with low cooperation. Figure 9 also shows positive impacts to the users of the system, making problematic stations less likely.

The small improvements observed are in contrast to other studies, like [1]. This can be explained by the low number of stations available at short distances (hence the jump at 700 metres). Also, a tendency of people moving to the north can be observed in Figure 8, resulting in the northern stations forming empty clusters. In this case, and also for single remote stations, an incentive-based strategy can only evenly distribute the load, but never completely compensate for the imbalance. "Connecting" those clusters by adding intermediary stations could improve this situation. Thus, as the system continues to grow, the impact of this strategy will likely increase.

Next, an optimal static redistribution is considered for comparison (**Static LD, Opt**). This is implemented in the model by resetting the number of available bikes to the optimum every day at midnight. The fill levels start at their optimum, as if the strategy was also used at the day before the first day in the simulation. The redistribution effort is calculated as 0 in this case, but will always be the same as for **Baseline LD, Opt** on day one. One can see, that although its performance is a lot better than for **Inc**, the redistribution effort is comparatively very high. Also, the incentive strategy could help to improve problematic stations throughout a day, as indicated by Figure 9.

To test this, the optimal static redistribution is combined with an incentive strategy (**Static LD, Opt, Inc**). We choose 700 meters for the radius and 25% for the cooperation factor, since those seem to offer a good tradeoff between user satisfaction and cost. This strategy achieves almost complete satisfaction and a very high number of trips, while only requiring half of the redistribution effort of the static only strategy and only incentivizing 22% of trips. Admittedly, a static redistribution will never be completely optimal, but the incentive strategy should also compensate for that.

6 CONCLUSION

In this work, we have evaluated the performance of a comparatively small BSS using model checking with SSTL. We introduced a CARMA model to extend the system, showed how SSTL can be used to evaluate model outputs and evaluated the performance of the BSS again, also considering an estimate of the latent demand. Finally, we compared different strategies for redistribution, considering the role of users as replacement or support for traditional strategies. Although ECH is used as an example, this approach can also be applied to other BSS. Some of the outcomes should even be applicable to comparable smaller systems.

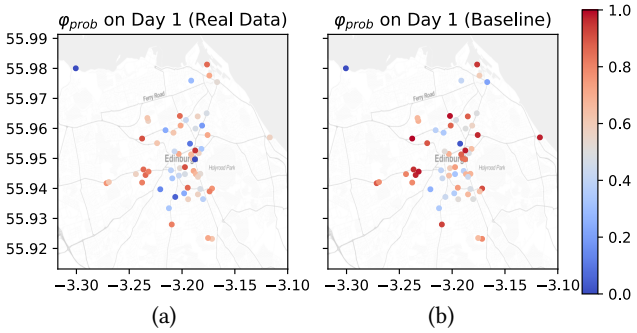


Figure 7: Satisfaction of φ_{prob} over 1 Day for the real data (a) compared to the baseline (b). Note that the overall trend is the same, but the baseline is saturated more because there are no redistributions.

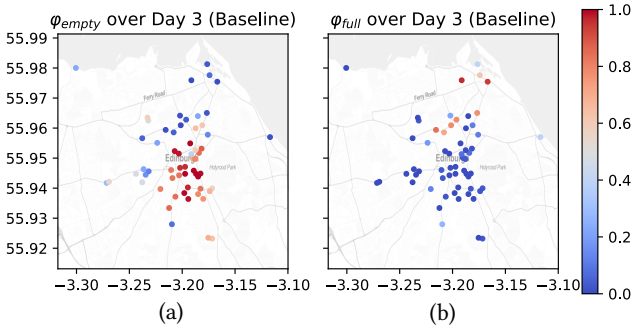


Figure 8: Satisfaction of φ_{empty} (a) and φ_{full} (b) on Day 3 for Baseline. Since there are no redistributions, this is exaggerated, but useful to show large scale behaviour. One can clearly see that there is a tendency for people to move north.

6.1 Discussion of the Used Techniques

In our case study, we apply SSTL for statistically checking the spatio-temporal behaviour. To the best of our knowledge, our study is the first to use SSTL in combination with a concrete simulation study conducted in collaboration between stakeholders and researchers. In this context we have found the combination of logic and simulation to be particularly powerful and identify the following specific benefits:

- **Interrogating time series and simulation results in a consistent manner.** The same formulas can be evaluated to query outputs of the model, historical logging data, and even real time data. This can be achieved while using the same or only slightly modified code. Starting from the same queries and using the same codebase means that the output format is also unified. Thus the results are directly comparable. In our study this is reflected in figures like Figure 9, where the satisfaction of φ_{prob} for the model outputs is directly comparable to the satisfaction for the real data. Going forward the same formulas could be used for monitoring the system in real-time allowing direct comparison of live data with historical and predicted performance.

Table 2: Comparison of the different redistribution strategies to the baseline. The intervals show the standard error of the simulation results. The abbreviations used are: "LD" for latent demand, "Opt" for optimal initial fill levels and "Inc d c" for the incentive strategy with maximum distance d and user cooperation factor c. For the incentive strategy used for "Static LD, Opt, Inc", $d = 700$ and $c = 0.25$. The incentive strategy seems to scale well with user cooperation, but not with distance. This is a consequence of the low number of stations. However, it can increase the number of trips made while lowering the amount of static redistribution that has to be done. Keep in mind that the maximum possible value for user dissatisfaction is 6. The mean and max are taken over all 3 days, which means slight changes can have a large impact.

Strategy		Redistribution Effort			No. of Incentives			No. of Trips	User Dissatisfaction $D(t)$	
		Day 1	Day 2	Day 3	Get	Ret.	Sum		Mean	Max
Baseline	–	40	51	56	–	–	–	1267±1.5	0.819	2.537
	LD	49	62	65	–	–	–	1355±1.7	1.109	2.853
	LD,Opt	40	56	62	–	–	–	1540±1.6	0.776	2.200
Static	LD,Opt,Inc	24	24	24	310±0.6	107±0.4	417	1860±1.4	0.095	0.248
	LD,Opt	40	40	40	–	–	–	1804±1.3	0.176	0.520
Inc 700	1.0	12	20	27	1020±1.4	271±1.1	1291	1736±1.4	0.267	0.833
	0.5	19	27	36	521±0.8	174±0.5	695	1725±1.4	0.309	0.886
	0.25	24	36	47	267±0.6	99±0.4	366	1670±1.5	0.440	1.041
Inc 600	1.0	23	40	55	632±1.6	185±0.8	817	1596±1.5	0.592	1.677
	0.5	25	39	53	364±1.0	119±0.5	483	1599±1.6	0.594	1.636
	0.25	27	43	55	206±0.6	73±0.4	279	1591±1.6	0.612	1.666
Inc 500	1.0	27	39	55	533±1.3	144±0.7	677	1576±1.5	0.629	1.774
	0.5	27	39	54	307±0.8	90±0.5	397	1577±1.5	0.632	1.842
	0.25	30	44	57	175±0.5	55±0.3	230	1574±1.6	0.647	1.798

- **Requirements of the simulation study are made explicit.** Requirements are closely related to the research questions and these are captured explicitly in the SSTL formulas. This means that it is clear, for example, whether space needs to be considered, whether a simulation model should be able to reproduce specific real world data, or a simulation model should give answers to specific questions that refer to the behaviour of the system [5]. Making behavioural requirements explicit in a formal language facilitates the discussion with stakeholders as well as reusing requirements throughout the simulation study [35]. As shown, SSTL offers an expressive and clear language to state such requirements and to probe the simulation model referring to various spatio-temporal dynamics based on a finite weighted undirected graph.

Also the use of CARMA for specifying the simulation model and producing the simulation trajectories proved beneficial. With CARMA's clear separation of concerns, the simulation model is accessible in a declarative, formal language [18], and could be specified succinctly which facilitated revising the model, and will facilitate extending it. In particular the separation of the behaviour aspect from the physical environment in which the BSS operates means that it is rapid and straightforward to consider changes in structural configuration of the system.

6.2 Lessons Learned from the Case Study

The first important result is that, in the current state of the ECH system, an incentive-based strategy alone is not enough to balance the system. The best results are seen when a 700 metre radius is used as the basis for incentives, but this is unreasonable in most cases. We believe that adding more stations could help reduce that radius to a more reasonable distance. As the system grows, this study can be repeated to test this hypothesis.

Whilst not completely solving the rebalancing problem, the incentive strategy does have positive impacts on the system. First, it can cut the effort needed for the (more expensive) static repositioning in half, even making it possible to only redistribute bikes over longer periods. Secondly, it helps to keep stations unproblematic throughout the day, increasing the likelihood of new customers finding a slot or bike where otherwise there would be none, keeping the satisfaction of the users high. When combining the static with the incentive-based repositioning, a high user satisfaction can be achieved. Also, the number of trips is increased, resulting in a more profitable system. To conclude, deploying an incentive-based strategy in Edinburgh has positive effects for users as well as the operator.

6.3 Future Work

The relatively short operational age of the system means that it is not currently possible to look at seasonal differences. Once the system has evolved, a new model that also accounts for differences

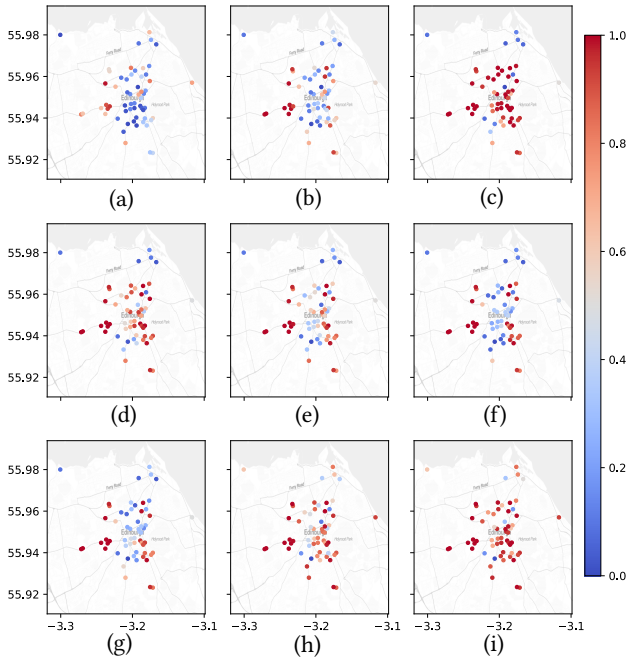


Figure 9: from (a) to (i): φ_{prob} for day two of Baseline LD, Baseline LD Opt, Inc 700 1.0, Inc 700 0.5, Inc 700 0.25, Inc 600 0.5, Inc 500 0.5, Static LD Opt, Static LD Opt + Inc. The abbreviations are the same as in Table 2.

in weekdays and seasons may be built. The influence of other factors, such as weather, could also be considered, as well as including more explicit information about the demands generated by festival attendees in August, available through ticketing data.

Furthermore, the operator of the ECH system is planning to introduce e-bikes in 2020. These will bring new challenges, such as depleted batteries and facilitating charging, that will need to be accounted for. Also up- and downhill trips will play a larger role when recharging by a dynamo is used meaning that a two-dimensional representation of space may be insufficient. Future work might thus include expanding the model to test the impact of e-bikes on the system and propose solutions to support their operation. Such a model could also include effects of dissatisfied users abstaining from using the system.

ACKNOWLEDGMENTS

We thank the operator of ECH for their close cooperation and for sharing the data. Special thanks go to Joshua Ryan-Saha and Ioanna Lampaki for arranging the necessary agreements and legal matter and for their general support. We thank Laura Nenzi for her support in using SSTL. Justin Kreikemeyer acknowledges the funding of the ERASMUS+ Programme and wants to express his gratitude to the University of Edinburgh for their hospitality.

REFERENCES

[1] Philipp Aeschbach, Xiaojing Zhang, Angelos Georghiou, and John Lygeros. 2015. Balancing bike sharing systems through customer cooperation - a case study on London's Barclays Cycle Hire. In *54th IEEE Conference on Decision and Control, CDC 2015, Osaka, Japan, December 15-18, 2015*. IEEE, 4722–4727. https://doi.org/10.1109/CDC.2015.7402955

[2] Szymon Albiński, Pirmin Fontaine, and Stefan Minner. 2018. Performance analysis of a hybrid bike sharing system: A service-level-based approach under censored demand observations. *Transportation Research Part E: Logistics and Transportation Review* 116 (2018), 59–69. https://doi.org/10.1016/j.tre.2018.05.011

[3] Yehia Abd Alrahman, Rocco De Nicola, and Michele Loreti. 2016. On the power of attribute-based communication. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*. Springer, 1–18.

[4] Panagiotis Angeloudis, Jun Hu, and Michael G. H. Bell. 2014. A strategic repositioning algorithm for bicycle-sharing schemes. *Transportmetrica A: Transport Science* 10, 8 (2014), 759–774. https://doi.org/10.1080/23249935.2014.884184 arXiv:https://doi.org/10.1080/23249935.2014.884184

[5] Osman Balci. 2012. A life cycle for modeling and simulation. *Simulation* 88, 7 (2012), 870–883. https://doi.org/10.1177/0037549712438469

[6] Nitin R Chopde and Mangesh K Nichat. 2013. Landmark based shortest path detection by using A* and Haversine formula. *International Journal of Innovative Research in Computer and Communication Engineering* 1, 2 (2013), 298–302.

[7] Hangil Chung, Daniel Freund, and David B Shmoys. 2018. Bike Angels: An Analysis of Citi Bike's Incentive Program. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*. 1–9.

[8] Claudio Contardo, Catherine Morency, and Louis-Martin Rousseau. 2012. *Balancing a dynamic public bike-sharing system*. Technical Report CIRRELT-2012-09.

[9] Sharon Datner, Tal Raviv, Michal Tzur, and Daniel Chemla. 2019. Setting inventory levels in a bike sharing network. *Transportation Science* 53, 1 (2019), 62–76.

[10] ECH. [n.d.]. Open Data - Edinburgh Cycle Hire. https://edinburghcyclehire.com/open-data. Accessed 01.01.2020.

[11] Elliot Fishman. 2016. Bikeshare: A review of recent literature. *Transport Reviews* 36, 1 (2016), 92–113. https://doi.org/10.1080/01441647.2015.1033036

[12] Christine Fricker and Nicolas Gast. 2016. Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *EURO J. Transportation and Logistics* 5, 3 (2016), 261–291. https://doi.org/10.1007/s13676-014-0053-5

[13] Vashti Galpin, Anastasis Georgoulas, Michele Loreti, and Andrea Vandin. 2018. Statistical Analysis of CARMA Models: An Advanced Tutorial. In *Proceedings of the 2018 Winter Simulation Conference (WSC '18)*. IEEE Press, 395–409.

[14] Nicolas Gast, Guillaume Massonnet, Daniël Reijsbergen, and Mirco Tribastone. 2015. Probabilistic Forecasts of Bike-Sharing Systems for Journey Planning. In *The 24th ACM International Conference on Information and Knowledge Management (CIKM 2015)*. https://doi.org/10.1145/2806416.2806569

[15] Daniel T Gillespie. 1976. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of computational physics* 22, 4 (1976), 403–434.

[16] Nanjing Jian, Daniel Freund, Holly M Wiberg, and Shane G Henderson. 2016. Simulation optimization for a large-scale bike-sharing system. In *2016 Winter Simulation Conference (WSC)*. IEEE, 602–613.

[17] Mathias John, Cédric Lhoussaine, Joachim Niehren, and Adelinde M Uhrmacher. 2010. The attributed pi-calculus with priorities. *Transactions on Computational Systems Biology* XII (2010), 13–76.

[18] Justin Kreikemeyer. [n.d.]. Code for this Paper. https://doi.org/10.5281/zenodo.3702267.

[19] Hon-Shiang Lau and Amy Hing-Ling Lau. 1996. Estimating the demand distributions of single-period items having frequent stockouts. *European Journal of Operational Research* 92, 2 (1996), 254–265.

[20] Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. 2015. Traffic Prediction in a Bike-Sharing System. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '15)*. Association for Computing Machinery, New York, NY, USA, Article Article 33, 10 pages. https://doi.org/10.1145/2820783.2820837

[21] Jenn-Rong Lin and Ta-Hui Yang. 2011. Strategic design of public bicycle sharing systems with service level constraints. *Transportation Research part E: Logistics and Transportation Review* 47, 2 (2011), 284–294. https://doi.org/10.1016/j.tre.2010.09.004

[22] Michele Loreti. [n.d.]. CARMA GitHub Repository. https://github.com/Quanticol/CARMA.

[23] Michele Loreti. [n.d.]. jSSTL Java Implementation. https://github.com/Quanticol/jsstl/.

[24] Michele Loreti and Jane Hillston. 2016. *Modelling and Analysis of Collective Adaptive Systems with CARMA and its Tools*. Springer International Publishing, Cham, 83–119. https://doi.org/10.1007/978-3-319-34096-8_4

[25] Oded Maler and Dejan Nickovic. 2004. Monitoring Temporal Properties of Continuous Signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, 22-24, 2004, Proceedings (Lecture Notes in Computer Science)*, Yassine Lakhnech and Sergio Yovine (Eds.), Vol. 3253. Springer, 152–166. https://doi.org/10.1007/978-3-540-30206-3_12

[26] Rahul Nair and Elise Miller-Hooks. 2011. Fleet management for vehicle sharing operations. *Transportation Science* 45, 4 (2011), 524–540. https://doi.org/10.1287/trsc.1100.0347

[27] Laura Nenzi, Luca Bortolussi, Vincenzo Ciancia, Michele Loreti, and Mieke Massink. 2017. Qualitative and Quantitative Monitoring of Spatio-Temporal Properties with SSTL. *CoRR* abs/1706.09334 (2017). arXiv:1706.09334 http://arxiv.org/abs/1706.09334

[28] Laura Nenzi, Luca Bortolussi, Vincenzo Ciancia, Michele Loreti, and Mieke Massink. 2018. Qualitative and Quantitative Monitoring of spatio-temporal Properties with SSTL. *Logical Methods in Computer Science* 14 (2018), 1–38.

[29] Laura Nenzi, Luca Bortolussi, and Michele Loreti. 2017. JSSTL - A Tool to Monitor Spatio-Temporal Properties. In *Proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'16)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL, 74–79. https://doi.org/10.4108/eai.25-10-2016.2266978

[30] Eoin O'Mahony and David B. Shmoys. 2015. Data Analysis and Optimization for (Citi)Bike Sharing. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, Blai Bonet and Sven Koenig (Eds.). AAAI Press, 687–694. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9698

[31] Samarth J Patel, Robin Qiu, and Ashkan Negahban. 2018. Incentive-based rebalancing of bike-sharing systems. In *INFORMS International Conference on Service Science*. Springer, 21–30.

[32] Julius Pfrommer, Joseph Warrington, Georg Schildbach, and Manfred Morari. 2014. Dynamic vehicle redistribution and online price incentives in shared mobility systems. *IEEE Transactions on Intelligent Transportation Systems* 15, 4 (2014), 1567–1578. https://doi.org/10.1109/TITS.2014.2303986

[33] Thomas Preisler, Tim Dethlefs, and Wolfgang Renz. 2016. Self-organizing redistribution of bicycles in a bike-sharing system based on decentralized control. In *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 1471–1480.

[34] Tal Raviv and Ofer Kolka. 2013. Optimal inventory management of a bike-sharing station. *IIE Transactions* 45, 10 (2013), 1077–1093. https://doi.org/10.1080/0740817X.2013.770186

[35] Andreas Ruscheinski, Tom Warnke, and Adelinde M Uhrmacher. 2019. Artifact-based Workflows for Supporting Simulation Studies. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[36] Jasper Schuijbroek, Robert Hampshire, and Willem-Jan van Hoeve. 2013. *Inventory rebalancing and vehicle routing in bike sharing systems*. Technical Report 2013-E1. Tepper School of Business, Carnegie Mellon University.

[37] Adish Singla, Marco Santoni, Gábor Bartók, Pratik Mukerji, Moritz Meenen, and Andreas Krause. 2015. Incentivizing Users for Balancing Bike Sharing Systems. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, Blai Bonet and Sven Koenig (Eds.). AAAI Press, 723–729. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9942

[38] Ji Won Yoon, Fabio Pinelli, and Francesco Calabrese. 2012. Cityride: a predictive bike sharing journey advisor. In *13th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 306–311. https://doi.org/10.1109/MDM.2012.16